Jessica Guo, Christine Do
CS214: Systems Programming

Assignment01: A better malloc() and free() due October 14, 2016

mymalloc.c implementation

This assignment used a static char array, myblock[5000] to simulate the heap. This is where we will allocate the dynamic memory called up by mymalloc(). Another void* array is initialized to hold the memory entries requested by the user (the metadata). All elements are initialized to 0, signifying that it is not in use. When mymallo() is called, a section of space is given corresponding to the requested size. Preceding the space in memoryNode struct head which helps keep track of how much space is used or available. mymalloc() will return the pointer, the section of memory, and the pointer to the memoryNode struct is stored in the memPtr array. When free() is called, the size of memoryNode is subtracted from the address contained in the pointer. It is then check against the memPtr for validity. If it is a valid pointer, then the space is freed and merged with adjacent free blocks. If the pointer is not valid, an error message is returned.

It detects all errors specified in the assignment01 PDF.

As the number of time mmalloc() is increases, the memoryNodes that are required for us to iterate through to find a block increase proportionally. This holds for myfree() as well; therefore, mymalloc() and myfree() have a time complexity of O(n).

Memgrind.c

To test any of the cases in memgrind.c. input a number 1-6 when prompted corresponding to:

1. A: malloc() 1 byte 3000 times, then free() 3000 1 byte pointers one by one
2. B: malloc() 1 byte and immediately free it 3000 times in a row
3. C: Randomly choose between a 1 byte malloc() or free() 6000 times
4. D: Randomly choose between a randomly sized malloc() or free 6000 times
5. E: Work load 1
6. F: Work load 2

Analysis of each workload:

A: malloc will be successful 200 times and return errors following. The average time is: 214.538208. The long time was probably due to the printing of the error messages
B: The average time was 0.076840. This is the fastest which is likely due to the fact that the area is being accessed each time malloc and free is called. There is no need to iterate to find free space.
C: The average time was 50.3360
D:The average time was 52.45900
The times in case C is more consistent and are generally around the same whereas case D has varied average time. This is probably due to the random sizes which can cause longer times.